JLO SAFE V2.4 has several more commands and functions added since the V2.2 command list enclosed was printed. I will detail these below:

FOR /n TO m or FOR /m ;This new command sets the start and range of a new fast integer only FOR-NEXT. n and m must be in the range of 0-65535. If m is set to 65535, the machine will loop forever. A STEP cannot be used (always +1) and no nested loops are allowed (only one SAFE LOOP at a time, but they CAN be nested with the slower Basic FOR/NEXT loops). The control variable name used is as set via use of a LET X=n at the very beginning of a Basic program or directly after a CLEAR command (IE: The simple (single letter) variable MUST be stored at the very beginning of the variables area). If the FOR /n TO m command is executed and a LET command has not been executed at the very start of the program, the computer will return with a 'FOR/ w/o LET' error. If the variable declared in the LET statement becomes negative by Basic manipulation, the computer will return with a 'neg FOR/ variable' error on encountering the next 'NEXT' statement. If the variable declared in the LET statement becomes a floating point variable by Basic manipulation, the computer will return with a 'fp FOR/ variable' error on encountering the next 'NEXT' statement. This new FOR/NEXT loop is desirable to use in place of a regular Basic FOR/NEXT loop where speed is important. The execution time of the new SAFE FOR/NEXT loop is 9 to 50 (or possibly even more at the end of really long programs) times faster than a regular FOR/NEXT loop, depending on its placement in the Basic program. (IE: A regular Basic loop gets slower the further it is within the program while this new looping structure is position independant speed-wise). A TS2068 computer using this new FAST SAFE FOR/NEXT loop can loop faster than the QL computer or even a Commodore 128 in FAST mode! If the 'n TO' is left off the command, the loop will start by setting the control variable to 1 (IE: FOR /1 TO 255 is identical to FOR /255 in operation) EG: FOR /2 TO 220 FOR /0 TO 100 FOR /1000 FOR /start TO end FOR /m-2 TO m+5 FOR x+x

NEXT ;This is the looping statement for the new FOR-NEXT loop described above. No arguments are allowed. Will cause control of Basic program to be passed to the next statement after the FOR / command if the control variable has not exceeded its limit as set in the FOR / command. EG: NEXT

EXAMPLES OF THE NEW FOR-NEXT COMMAND IN BASIC:

Causes 700 Xs to be printed on the screen;

```
 5 LET a=0
10 FOR /700: PRINT"X";:NEXT
```

Causes the numbers 0 through 21 to be printed in center of screen;

```
 5 LET L=0
10 FOR /0 TO 21: PRINT AT L,15;L
20 NEXT
```

Cause the INKEY$ function to be executed VERY fast till a key is pressed;

```
 5 LET S=PI: FOR /65535
10 IF INKEY$="" THEN NEXT
20 RETURN
```

The new fast SAFE loop will not work on a Basic AROS cartridge unless the FOR / and NEXT are both on the same Basic line, possibly not even then (untested).

MERGE /"FILENAME" ;This command is a simplified but VERY fast Basic program merge. Using this command instead of LOAD will append a Basic program with its variables (if any are present) onto the end of the current Basic program, without erasing the current program or its variables. This command will ONLY append the merged program and NOT insert lines or delete current lines or variables. The merged program should be written with high line numbers with this restriction born in mind. This command can be used within a running program, along with use of the regular 2068 DELETE command, to make far larger Basic programs than are normally possible by merging subroutines in as needed, using them, deleting them, merging more as nec., etc.. Room is dynamically opened for the merged code and loaded directly in so speed is VERY fast even for very long subroutines. Variables merged are stored at the beginning of the variables area, so beware that if a new variable is merged with a Basic program that has the same name as a variable used by the main program, Basic will find and use the merged variable rather than the main variable. The main program variable will still be in memory, but Basic will not be able to access it. Beware also that if merged code has lower line numbers than the main Basic program, Basic is very likely to become confused and stop with an error. The limitation that the merged code have higher line numbers than tha main program's code was considered a good trade for the very much increased speed of loading the new

code in within a running program, because the size of programs merged using a disk system in this way tends to increase making the time required for Basic to actually insert & delete lines as is normally done with a cassette merge simply too long to be really pratical. EG: MERGE /"9000 gosub" MERGE /"menue#1" MERGE /G$

SAVE //"FILENAME" ;This variation of the SAVE command using two "/" instead of only one tells SAFE to save a file without a "FILE EXISTS! 5 SECONDS TO ABORT" message and its "TOOT" w/delay. The idea is to use this variation of the SAVE command when you already know or expect the filename used to be present on the disk, and wish to overwrite it with the delay and warnings that would occur in doing so using the regular SAVE / command. This is most often needed on a program such as a BBS that is running unattended and constantly updating files. Situations like this do not need this warning or the delay it causes, so this command gives the programmer a way around them. The LOAD & MERGE commands will also accept two "/" after the initial token, but no change in their operation will result in two being there. EG: SAVE //"userbase" DATA U$() SAVE //"EMAIL" DATA E$() SAVE //"newtext" CODE 32768,10000 SAVE //G$ DATA Z()

New "functions" added are:

An NMI SCREEN$ save. A press of keys Q,W,E,R, or T after a press of the NMI pushbutton causes the current screen to be saved to disk as a BYTES file whose name is A,B,C,D or E respectively. The name of the SCREEN$ file can later be changed as desired from Basic using the RESTORE /"FN"CODE TO "NFM" command. As usual for SCREEN$, only the regular 32 column display file is SAVED, not the 64 column screen.

RESET. Holding the "N" key and then pressing the NMI SAVE button will cause a computer reset to occur. Both Basic and SAFE will totally re-initialise just as if the power switch were turned off then back on, but with much less chance of a power-up "glitch" corrupting data on a disk installed in one of the drives. The addition of this function is almost like adding a reset switch without physically adding one!